I Claim:

1. A method for debugging a computer system, comprising:

initiating a process in the computer system, the process including instructions;

launching a debugger program that is embedded in a ROM of the computer system;

executing at least part of the instructions; and

the debugger program operating on at least part of the executed instructions.

2. A method as defined in claim 1, further comprising:

the debugger program capturing a trace of the at least part of the executed instructions.

3. A method as defined in claim 2 wherein:

the computer system includes a port connected to a monitoring system; and further comprising:

outputting the captured trace through the port to the monitoring system.

4. A method as defined in claim 1, wherein:

the process comprises a boot process;

the instructions comprise boot instructions stored in the ROM of the computer system; and

the debugger program is launched from within the boot process.

5. A method as defined in claim 1, further comprising:

stopping the execution of the instructions at a first break point;

the debugger program setting a second break point in the instructions; and

continuing the execution of the instructions.

6. A method as defined in claim 5, further comprising:

the debugger program disassembling a current instruction of the instructions;

determining a length of the current instruction, the length of the current instruction indicating a start point for a next instruction of the instructions; and

setting the second break point at the start point of the next instruction.

7. A method as defined in claim 1, further comprising:

interrupting the execution of the instructions at a current instruction;

the debugger program operating on the current instruction by disassembling the current instruction; and

executing the current instruction.

8. A method as defined in claim 7, further comprising:

before executing the current instruction, determining a location of a next instruction of the instructions; and

after executing the current instruction, interrupting the execution of the instructions at the next instruction.

9. A method as defined in claim 1, further comprising:

setting a switch within the computer system; and

launching the debugger program in response to detecting the set switch.

10. A method as defined in claim 1, further comprising:

setting a break point within the process according to a location specified in a map of the process contained in the debugger program.

11. A computer system, comprising:

at least one processor;

a read-only memory (ROM) connected to the processor;

a target process having instructions capable of being executed by the processor; and

a debugger program embedded within the ROM and capable of being executed by the processor to operate on at least part of the instructions of the target process.

12. A computer system as defined in claim 11, wherein:

the target process comprises a boot process.

13. A computer system as defined in claim 11, wherein:

the debugger program operates on the at least part of the instructions by capturing a trace of the execution of the at least part of the instructions.

14. A computer system as defined in claim 11, wherein:

the debugger program further operates on the at least part of the instructions by disassembling the at least part of the instructions.

15. A computer system as defined in claim 11, wherein:

the debugger program operates on a current instruction of the target process and sets a break point at a next instruction of the target process to be operated on.

16. A computer system as defined in claim 11, further comprising:

a switch that when set enables launching of the debugger program;

and wherein the target process launches the debugger program upon detecting that the switch is set.

17. A computer system as defined in claim 11, further comprising:

an interrupt flag that when set causes an interruption of execution of the target process;

and wherein, upon interruption of the execution of the target process, the debugger program operates on a current instruction of the target process.

18. A computer system as defined in claim 11, further comprising:

a map of the target process embedded in the debugger program and specifying locations of parts of the target process.

19. A computer debugging system, comprising:

a target computer;

a monitoring system connected to the target computer;

a data storage device in the monitoring system;

a read-only memory (ROM) in the target computer;

a target process having instructions executable in the target computer; and

a debugger program embedded in the ROM and executable in the target computer to generate data on the execution of at least part of the instructions of the target process and to transfer the data to the monitoring system for recording in the data storage device.

20. A computer debugging system as defined in claim 19, further comprising:

a disassembler embedded in the ROM and executable in the target computer to disassemble the at least part of the instructions of the target process.

21. A computer debugging system as defined in claim 19, wherein:

the data on the execution of the at least part of the instructions of the target process comprises a trace capture of the at least part of the instructions.

22. A computer system, comprising:

a read-only memory (ROM) means for storing computer control instructions;

a means for executing a target process;

a ROM-embedded means for interrupting the execution of the target process at a current instruction;

a ROM-embedded means for disassembling the current instruction;

a means for executing the current instruction; and

a ROM-embedded means for capturing a trace of the current instruction and of results of the execution of the current instruction.

23. A computer system as defined in claim 22, further comprising:

a means for transferring the captured trace to an external means for storing the captured trace.

24. A computer system comprising:

a read-only memory (ROM);

a target process having executable instructions; and

a debugger program embedded within the ROM and having a disassembler and a trace capturer;

and wherein:

the debugger program interrupts execution of the target process at some of the instructions;

the disassembler disassembles at least some of the instructions at which the execution of the target process is interrupted; and

the trace capturer captures a trace of at least some of the disassembled instructions.

25. A computer system as defined in claim 24, wherein:

the target process comprises a boot process stored within the ROM and having instructions that boot the computer system when the boot process executes.

26. A computer system as defined in claim 24, further comprising:

a port that can be connected to a monitoring system;

and wherein the debugger program outputs the captured trace through the port to the monitoring system.

27. A computer system as defined in claim 24, wherein:

the debugger program disassembles a current instruction of the target process, determines a length of the current instruction, and sets a break point at a next instruction of the target process.

28. A computer system as defined in claim 24, wherein:

the computer system executes the current instruction, encounters the

break point at the next instruction, and jumps to the debugger program with the next instruction as a new current instruction.

29. A computer system comprising:

a switch;

a target process having executable instructions; and

a debugger program;

and wherein:

when the switch is off, the debugger program cannot be launched; and

when the switch is on, the debugger program can be launched to interrupt execution of the target process at some of the instructions and operate on at least some of the instructions at which the execution of the target process is interrupted.

30. A method for debugging a target process executing on a computer system, comprising:

launching a debugger program from a read-only memory (ROM) of the computer system, the ROM having a boot process and the debugger program embedded therein, the debugger program having a disassembler and a trace capturer;

interrupting execution of the target process at an instruction;

the disassembler disassembling the instruction; and

the trace capturer capturing a trace of the instruction.

31. A method as defined in claim 30, further comprising:

executing the instruction; and

interrupting the execution of the target process after the instruction.

32. A method as defined in claim 30, wherein:

the target process comprises the boot process.

33. A method for debugging a target process executing on a computer system, comprising:

setting a switch within the computer system to one of an on state and an off state;

when the switch is set to the off state, preventing execution of a debugger program; and

when the switch is set to the on state:

launching the debugger program;

interrupting execution of the target process at an instruction; and

the debugger program operating on the instruction.